UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/693,717	10/24/2003	Holly Knight	MS306958.1/MSFTP545USA	1585

27195 7590 11/10/2004

AMIN & TUROCY, LLP
24TH FLOOR, NATIONAL CITY CENTER
1900 EAST NINTH STREET
CLEVELAND, OH 44114

EXAMINER	
CHOW, CHIH CHING	

ART UNIT	PAPER NUMBER
2122	

DATE MAILED: 11/10/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/693,717	KNIGHT ET AL.	
	Examiner	Art Unit	
	Chih-Ching Chow	2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *24 October 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *24 December 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *08/19/04*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed on October 24, 2003.

2. The priority date considered for this application is October 24, 2003.

3. Claims 1-24 have been examined.

Specification

4. The lengthy specification has not been checked to the extent necessary to determine the presence of all possible minor errors. Applicant's cooperation is requested in correcting any errors of which applicant may become aware in the specification.

Claim Objections

5. Claim 24 is objected to because of the following informalities: 2nd line, 'carry out' should be 'carrying out'. Appropriate correction is required.

Claim Rejections - 35 USC § 112

6. The following is a quotation of the **first paragraph** of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention. *** 112 (1st)

7. Claims 1, 12 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter

which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

Claim 1 recites: "A system for extending application preference classes comprising"

Claim 12 recites: "A method for extending application preference class functionality comprising...."

It is not clear to the examiner what does 'extending application preference classes' mean. The word 'preference' has been recited many places and in many different ways throughout the entire specification, e.g. 'preference execution engine', 'preference evaluation', 'preference class', 'write preferences to the data store', 'tables of events and preferences', 'end-user preferences', 'valid preference', 'preference authoring', 'preference group',.. etc, it's defined on page 14, 'Preferences are end-user defined rule that are triggered by the occurrence of events', and first line of page 19, 'A preference class is a unit of information agent schema development. A preference includes a set of allowed condition classes (e.g., IsFrom (X), IsTo(Y)) and action classes (e.g., MoveToFolder (Z), Delete()). Furthermore, every preference is associated with a specific event class or trigger to initiate an action (e.g., EmailEvent).'; and on page 21, 2nd paragraph, 'Preferences

can be specified through a user interface (e.g. control panel, toolbar).' However it is still not clear of the manner and process of making and using the claimed feature. It was not clear **what the base 'application preference classes' are, and what are they 'extending' to?**

8. Claims 9-11 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

Claim 9 recites: "A system for extending conditions constants comprising:

(a) an input component for receiving a programmatic constant; and

(b) an accessory component for determining the constant value from data stored across a plurality of domains.

Claim 10 recites: "The system of claim 9, wherein the constant is first order constant."

Claim 11 recites: "The system of claim 9, wherein the constant is an Nth order constant."

It is not clear to the examiner what do 'extending conditions constants', 'first order constant', 'Nth order constant' mean. The word 'conditions' and 'constants' has been recited many places and in many different ways throughout the entire specification, the only close definition is on page 29, 'Constant accessors are very powerful constants that allow preferences and conditions to be written that are capable of navigating and retrieving information from various domains. The constants are simply names veneered over functions that operate to find and materialize the correct information, namely the members of the group associated with the name of the constant', and first line of page 31, lines 12-13, 'constant extensions are similar to conditions on fields of items that can also be represented as constant accessors and combined with other constants' and on page 31, 2nd paragraph, 'Constants discussed thus far are known as first order constants as they are defined relative to a given user. And accessor component 510 or accessor can then key off a user's identify or other starting points. It should also be noted that Nth order constants can also be composed and saved by a user by using preferences to combine previously defined groups.' However it is still not clear of the manner and process of making and using the claimed feature. **It is not clear what the base 'condition constants' are, and what they are 'extending' to?** As a

matter of fact, **if it's a 'constant', how can it be 'extended'**? - according to the definition of a constant, it is a named item that **retains a consistent value** throughout the execution of a program (see Microsoft Press Computer Dictionary, Third Edition 1997, page 114).

9. Claims 21-24 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

Claim 21 recites: "A method for extending programmatic constants comprising: receiving a programmatic constant; and resolving the value of the constant by searching across application domains."

Claim 22 recites: "The method of claim 21, wherein the constant is a first order constant."

Claim 23 recites: "The method of claim 21, wherein the constant is an Nth order constant."

It is not clear to the examiner what do 'extending programmatic constants', 'first order constant', 'Nth order constant' mean, see the description in item 7. **It is not**

clear what the base 'programmatic constants' are, and what they are 'extending' to? As a matter of fact, **if it's a 'constant', how can it be 'extended'**? - according to the definition of a constant, it is a named item that **retains a consistent value** throughout the execution of a program (see Microsoft Press Computer Dictionary, Third Edition 1997, page 114).

Claim 24 recites: "A computer readable medium having stored thereon computer executable instructions for carry out the method of claim 21". Since claim 21 is under 35.U.S.C. 112 (1st) rejection, it's not clear for claim 24 for what to 'carrying out', therefore claim 24 is also rejected.

10. The following is a quotation of the **second paragraph** of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

11. Claim 1(b) recites the limitation in "the register component', but 1(a) claimed 'an instance registry component'. Claim 1 (c) recites the limitation in "the instance registry". There is insufficient antecedent basis for this limitation in both claims 1 (b) and 1 (c). Examiner assumes claims 1 (b) and 1 (c) actually meant to be 'instance registry component', same as 1 (a) claimed.

Claim Rejections - 35 USC § 103

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

13. Claims 1-2, 6-8, 12-14 are rejected under 35 U.S.C. 103(a) as being unpatentable over 'IBM Ada/6000', September 02, 1998; (hereinafter "IBM"); and US Patent No. 6,026,235 by Steven T. Shaughnessy et al. (hereinafter "Shaughnessy"); in view of U.S. Patent No. 6,016,394 by Jeffrey L. Walker (hereinafter "Walker").

Claim	**IBM / Shaughnessy / Walker**
1. A system for extending application preference classes comprising: (a) an instance registry component; (b) a first application including one or more functions that are registered in the register component; and (c) an extension component which reads candidate functions from the instance registry and creates bindings between a second application and a first such that the second application can utilize the functions of the first application.	IBM Ada 6000 teaches a compiling environment which allows user to define a first application and further include second application (or more applications). To summarize the IBM Ada / 6000 compiler (refer to 'IBM Ada/6000' Synopsis): • Ada has packages which are independent computer software components. • "adalib" is the sublibrary refers to the Ada library which is the current working library • "alib.list" is crated whenever an

'alibinit' command is entered (a sublibrary 'adalib' is also created). 'alib.list' specifies the current working directory/directories (as long as they are accessible). User can specify program units located in other library/libraries that he/she would need to get the designated program compiled; in other words, user can include other libraries in the current library by editing the alib.list file and **register** all the file objects that he would need in order to get the designated program to be compiled with no error.

- obscure error message would be produced whenever the compiler confronts a library unit is missing during compiling.

For item (a), IBM's 'adalib' functioning as the **instance registry component**. It's also taught by Shaughnessy, see Shaughnessy column 2, lines 12-19, "In typical operation, a linker receives, either from the user or from an integrated **compiler, a list of object modules desired to be included in the link operation.** (*instance registry component*) The linker scans the object modules from the object and library files specified. After resolving interconnecting references as needed, the linker constructs an **executable** image by organizing the object code

from the modules of the program in a format understood by the operating system program loader." In column 3, lines 8-10, "In addition to the additional calls, the approach requires a special **executable** link operation **to bind** in the function entry/exit function calls and the required runtime support for them." For item (b), the application(s) including one or more functions are specified in the **alib.list**.

(c) All the program units been specified in the **alib.list** will be compiled and bound by the compiler. Therefore, the second application can be 'bound' with the first application's functions as long as it has the 'visibility' (accessible) to those functions. Here the '**extension component**' can be the *Ada compiler* itself. It has the visibility to all the files that has been compiled, and can do binding among them.

IBM and Shaughnessy teach the computer feature to include different application components in a computer compiling/binding environment, however they don't mention 'extending application preference classes' specifically, however, Walker teaches extending application preference for applications in an analogous prior art. In Walker, column 16, lines 50-56, "A prototype smart codes transaction 195 facilitates the **addition** *(extending)* of a new smart code transaction which creates a smart code set and may define **a new smart**

	pick using the smart code set. By using smart codes, default values may be specified as **preferences** for use in application creation. Smart codes provide a centralized location for **lists of values in a custom target application.**" Further more, column 4, lines 59-62, "In the present invention, the concept of "sets" allows an application designer to define all the data on which the target application will possibly operate." (extending *application preferences*). It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the feature of including multiple application units by extending the application preferences further taught by Walker for the purpose of creating a target software applications for end-users. (see Walker's Abstract line 1)..
2. The system of claim 1, wherein the instance registry component comprises a definition registry for storing function definitions and a binding registry for storing binding data.	For the feature of claim 1 see claim 1 rejection. The *adalib* is the '**instance registry component**' since adalib contains both 'definition registry' and 'binding registry'; it stores all the function definitions and the binding data.
6. The system of claim 1, wherein candidate functions are only available for binding to specific applications.	For the feature of claim 1 see claim 1 rejection. Functions are only available for binding if they are defined in an application, which are part of the 'to-be-compiled' list (alib.list), see IBM page 6

	'Including Other Libraries in Yours'.
7. The system of claim 2, wherein the binding registry receives function binding information from an extension data file (EDF).	For the feature of claim 2 see claim 2 rejection. Again, the 'alib.list' has the same function as the 'extension data file', since it contains all the binding registry information.
8. The system of claim 1, wherein the binding is broken upon removal of the function providing application.	For the feature of claim 1 see claim 1 rejection. An 'obsolete' error will be produced if any program component is missing from the adalib (say **removal** of a function), therefore the 'binding is **broken**'. See IBM page 7, 'Errors During A Compile or Bind'.
12. A method for extending application preference class functionality comprising: (a) receiving an extension data file (EDF) containing information about candidate function bindings; (b) registering one or more function bindings in a central data store; and (c) binding a function of a first application to a second application utilizing binding function information located in central data store.	Same as claim 1 rejection. Where alib.list is the **extension data file**; the **central data storage** is the adalib itself; the **binding** is done by the Ada compiler.
13. The method of claim 12, further comprising applying acceptance logic to determine whether the second application will accept the binding.	For the feature of claim 12 see claim 12 rejection. See page 10, 2nd paragraph of the Office Action.
14. A computer readable medium having stored thereon computer executable	The IBM example given in claim 1, an executable file (defaulted **a.out**, see

instructions for carrying out the method of claim 12.	IBM page 5, 1st paragraph) will be created after the **Ada** command is executed (if no compiling and binding error). The Ada compiler contains the **instructions carrying out the method of Ada compiling and binding**, the adalib itself **stores the executable** which is produced after compiling and binding.

14. Claims 1-8, 12-14, and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over 'IBM Ada/6000', September 02, 1998; (hereinafter "IBM"); and US Patent No. 6,026,235 by Steven T. Shaughnessy et al. (hereinafter "Shaughnessy"); in view of U.S. Patent No. 6,016,394 by Jeffrey L. Walker (hereinafter "Walker"); further in view of 'Compile Time Scheduling of an Ada Subset' by E.W. Giering III et al., Washington Ada Symposium Proceedings, June 1990 (hereinafter "Giering").

Claim	**IBM / Shaughnessy / Walker / Giering**
3. The system of claim 1, wherein the functions provide conditions.	For the feature of claim 1 see claim 1 rejection. IBM Ada compiler and Shaghnessy teach the feature of functions but does not mention 'conditions' specifically, however Giering teaches this feature in an analogous prior art. In Giering, page 153, section 6.1, 2nd paragraph, '**conditional statements** could be allowed in our

model. This would amount to an **implicit conditional compilation** feature'.

4. The system of claim 1, wherein the functions provide events.

For the feature of claim 1 see claim 1 rejection. IBM Ada compiler teaches the feature of functions but does not mention 'events' specifically, however Giering teaches this feature in an analogous prior art. In Giering, page 153, section 6.1, 3rd paragraph, 'Ada **tasks** (*events*) **can also be referenced** by pointer or kept in an array. Making or accepting **an entry call on a task** in such an array is another form of data-dependent execution of a rendezvous primitive'.

5. The system of claim 1, wherein the functions provide accessors.

For the feature of claim 1 see claim 1 rejection. IBM Ada compiler teaches the feature of functions but does not mention 'accessors' specifically, however Giering teaches this feature in an analogous prior art. In Giering, page 143, column2, 3rd paragraph, 'Since any **task** can **call an entry**' (*accessors*) - in Ada, making calls to task is implied.
It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the instance registry (compiling list), and IBM's alib.list feature by the including the conditions/ events/ accessors features further taught by Giering for the purpose of scheduling a restricted form when compiling a program (see Giering first paragraph).

15. Claims 15-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over 'IBM Ada/6000', September 02, 1998; (hereinafter "IBM"); and US Patent No. 6,026,235 by Steven T. Shaughnessy et al. (hereinafter "Shaughnessy").

Claim	IBM / Shaughnessy
15. A method of uninstalling an application comprising: (a) removing all application registrations from central storage location; (b) removing Program Components; and (c) notifying dependant applications.	**Uninstalling (removing)** any library unit from alib.list, the adalib will **remove** that specified unit from the working library, and all the dependent units will be recompiled. The **central storage location** is the adalib, it's also an instance registry component (see claim 1 (a) rejection). If any obsolete error has occurred, the **obscure error message** will be displayed on the compiling screen (*notifying dependent applications*). See IBM page 7, 'Errors During A Compile or Bind'.
16. The method of claim 15, wherein the central storage location is an instance registry.	For the feature of claim 15 see claim 15 rejection.
17. The method of claim 16, wherein the instance registry comprises a definition registry and a binding registry.	For the feature of claim 16 see claim 16 rejection. The adalib itself functions the same as '**instance registry**', which contains the definition registry and the binding registry (see claim 1 rejection).
18. The method of claim 17, wherein removing registrations comprises removing registrations in the definition	For the feature of claim 17 see claim 17 rejection. For the rest claim 18 features see claim 15 rejection.

registry and the binding registry.

19. The method of claim 15, wherein the notifying dependant applications causes dependant applications to place their dependencies in a NotAvailable state.	For the feature of claim 15 see claim 15 rejection. The 'NotAvailable' is the same as 'obsolete' state, therefore see 'obsolete error message' part of claim 15 rejection.
20. A computer readable medium having stored thereon computer executable instructions for carrying out the method of claim 15.	For the feature of claim 15 see claim 15 rejection. The IBM Ada compiler contains the **instructions carrying out the method of Ada compiling and binding** based on the alib.list, therefore any unit is removed from the alib.list will cause a removal of that unit from the adalib. Again, the adalib itself **stores the executable**, which is produced after compiling and binding.

Conclusion

The following summarizes the status of the claims:

35 USC § 112 (1st) claim rejection: claims 1, 9-12, 21-24

35 USC § 112 (2nd) claim rejection: claim 1

35 USC § 103 claim rejection: claims 1-8, and 12-20.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 703-305-7205. The examiner can normally be reached on 7:00am - 3:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam can be reached on 703-305-4552. The fax phone

number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow
Examiner
Art Unit 2122

CC

ANTONY NGUYEN-BA
PRIMARY EXAMINER